McGill University
School of Computer Science
COMP-202A Introduction to Computing 1

Final Exam – Version 1

Wednesday, December 20, 2006, 9:00-12:00 (GYM)
Instructors: Mathieu Petitpas, Shah Asaduzzaman, Sherif Shaker
Associate Examiner: Joseph Vybihal

Student Name: _____

Student ID: _____

Your Section: ☐ Mathieu Petitpas (Sect 1) ☐ Shah Asaduzzaman (Sect 2) ☐ Sherif Shaker (Sect 3)

Instructions

- This is a 3-hour final exam.
- Attempt all questions.
- Parts marks are given for all questions (except multiple choice questions). Show your work and do not leave a question empty.
- Write all answers in the exam booklet provided, except for the multiple choice questions, which you **MUST** answer on the questionnaire.
- You **MUST** return this questionnaire along with the exam booklet at the end of exam.
- No notebooks, calculators, or textbooks permitted in this exam.
- Language translation dictionaries are permitted.
- You are permitted to write your answers in either English or French.
- There is a Java Standard Library reference appended to the questionnaire. You may use this reference to write your answers

Grading

Section	Grade	Your Mark
Section 1: Multiple Choice	20	
Section 2: Problems	40	
Section 3: Programming	40	
Total	100 %	

Section 1: Multiple Choice Questions (2 points each; 20 points total)

Each of the following multiple choice questions has five choices labeled A-E and has **EXACTLY ONE** correct answer. You **MUST** write your answer (A-E) in the box below under the corresponding question.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10

1. Each byte of main memory in a computer is addressed using a unique number. You know that reference variables actually store the address of a memory location. If the storage size of a reference variable is 4 bytes, what is the maximum amount of memory the computer can have?

A) 1024 megabytes
B) 2^4 bytes
C) 10^4 bytes
D) 4 gigabytes
E) $(2^4)^4$ bytes

2. We often use the following Java statement before we start reading user input in our Java programs:

```
Scanner Keyboard = new Scanner(System.in);
```

Which of the following is TRUE regarding the above Java statement?

A) This statement creates a new `Keyboard` object and assigns that to a `Scanner` object.
B) `Keyboard` is a Java keyword / reserved word here, which sets the computer keyboard ready for user input.
C) `System.in` is a static member variable (attribute) of the `System` class that refers to the standard input stream.
D) After this statement a new object of class `Keyboard` is created.
E) This statement creates a new object which belongs to the `Scanner` class, using a constructor that takes an object named `Keyboard` as parameter.

3. If `a`, `b` and `c` are three integer variables, which one of the following is NOT a valid arithmetic expression in Java?

A) `(a > b) ? a : ((b > c) ? b : c))`
B) `c = a + 2*b - 45`
C) `a**2 + 2 * a * b + b**2`
D) `a+++b+c`
E) `a + b + -c`

4. What is the output of the following program segment?

```
for (int i=0; i<5; i++) {
     if (!(i >= 3 && i < 4))
          System.out.print(i + ",");
}
```

A) 0,1,2,3,4,5,
B) 0,1,2,4,
C) 0,1,2,3,4,
D) 0,1,2,5,
E) 0,1,2,

5. Which of the following four program segments involving iterations are equivalent to each other in terms of output?

1)
```
int i=0;
for (i=0; i<10; i++) {
     System.out.println(i);
}
```

2)
```
int i=0;
while (i < 10) {
     System.out.println(i);
}
i++;
```

3)
```
int i = 0;
while (i < 10) {
     i++;
     System.out.println(i);
}
```

4)
```
int i = 0;
while (i < 10) {
     System.out.println(i);
     i = i+1;
}
```

Select one:
A) 1, 2, 3, and 4
B) 2, 3, and 4
C) 1 and 4
D) 2 and 3
E) 1 and 3

6. Which of the following statements correctly creates an array of five empty `Strings`?

1. `String a[] = new String [5];`
   `for (int i = 0; i < 5; i++)`
   `a[i] = "";`
2. `String a[] = {"", "", "", "", ""};`
3. `String a[5];`
4. `String[5] a;`
5. `String[] a = new String [5];`
   `a = {"", "", "", "", ""};`

A) None
B) Only 1
C) Only 1 and 2
D) Only 1, 2 and 5
E) All of the above

7. Which of the following statements about arrays are FALSE?

1) Valid array indexes for an array variable called `myArray` are in the range between 1 and `myArray.length` inclusively.
2) Creating an array of object references (an array of `String` objects, for example) does not create the actual objects whose references will be stored in the array.
3) Arrays themselves are objects, so an array variable is a reference to the actual array; therefore, two different array variables can be used to refer to the same array in memory.
4) If `myArray` is an array variable, it is possible to make the array grow or shrink by changing the value of `myArray.length`.
5) A two-dimensional array is basically an array of array objects.

A) 1 and 3
B) 1 and 4
C) 2 and 4
D) 2 and 5
E) 3 and 5

8. Which of the following statements about `static` / `final` members are FALSE?

1) Local variables and parameters can be `final`, but they cannot be `static`.
2) The value of a variable declared as `final` cannot be changed after it has been set once.
3) A static method can refer to an instance member (variable or method) without using the name of an object variable as a prefix.
4) An instance method can refer to a static member (method or variable).
5) Each object that belongs to a class has its own copy of static variables, and the values of these copies can change independently of the values of other copies.

A) 1 and 3
B) 1 and 4
C) 2 and 4
D) 2 and 5
E) 3 and 5

9. Which of the following statements about exception handling are FALSE?

1) If a statement inside a method `m()` throws an exception and this method does not catch this exception, the exception is propagated to the method's caller.
2) A `try` block can only have one `catch` clause.
3) To catch an exception, one can write the statement that may cause the exception inside a `try` block with a `catch` clause for that exception type.
4) If a statement in a method throws a checked exception, you must either catch and handle it inside the method or add a `throws` clause to the method header.
5) `NullPointerException` and `ArrayIndexOutOfBoundsException` are checked exceptions.

A) 1 and 2
B) 1 and 4
C) 2 and 5
D) 3 and 4
E) 3 and 5

10. How many of the following statements are FALSE?

1. A constructor is a special method with the same name as the class that is used to initialize the members of a class object. Constructors are called when objects of their classes are instantiated.
2. A method declared `static` cannot access non-`static` members and can not invoke non-`static` methods directly. A `static` method cannot use the `this` reference because `static` class variables and `static` methods exist independent of any objects of a class.
3. An array subscript (index) may be an integer or an integer expression. If a program uses an expression as a subscript, then the expression is evaluated to determine the particular element of the array.
4. Overloaded methods can have different return values, and must have different parameter lists. Two methods differing only by return type will result in a compilation error.
5. To pass one row of a double-subscripted (two dimensional array) to a method that receives a single-subscripted array (single dimensional array), simply pass the name of the array followed by the row subscript (index).

A) 0
B) 1
C) 2
D) 3
E) 4

Section 2: Problems (10 points each; 40 points total)

1. (10 points) Consider the following program. What will be displayed when the `main()` method of class `Deceive` is executed?

```
class Point {
   private int x;
   private int y;

   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }

   public void set(int x, int y) {
      this.x = x;
      this.y = y;
   }

   public String toString() {
      return "(" + this.x + ", " + this.y + ")";
   }
}

public class Deceive {
   public static void trick(int i1, int i2, Point p1, Point p2, Point[] pa,
      int[] a1, int[] a2) {
      i1 = 11;
      i2 = 12;

      p1.set(13, 14);
      p2 = new Point(15, 16);
      pa[0] = new Point(17, 18);

      a1[0] = 19;

      a2 = new int[1];
      a2[0] = 20;
   }

   public static void main(String[] args) {
      int int1;
      int int2;
      int[] array1;
      int[] array2;
      Point point1;
      Point point2;
      Point[] pointArray;

      int1 = 1;
```

```
        int2 = 2;

        point1 = new Point(3, 4);
        point2 = new Point(5, 6);

        pointArray = new Point[1];
        pointArray[0] = new Point(7, 8);


        array1 = new int[1];
        array1[0] = 9;

        array2 = new int[1];
        array2[0] = 10;


        trick(int1, int2, point1, point2, pointArray, array1, array2);


        System.out.println(int1 + ", " + int2);
        System.out.println(point1 + ", " + point2 + ", " + pointArray[0]);
        System.out.println(array1[0] + ", " + array2[0]);
    }
}
```

2. (10 points) In a list of numbers, the number that occurs most often is called the *mode* of that list. For example, consider the following list of numbers:

2, 5, 6, 7, 7, 10, 11, 11, 13, 15, 15, 15, 19, 20, 20

The number which occurs most often in the above list is 15 (it occurs 3 times). Therefore, 15 is the mode of this list.

Write a method called `findMode()` that takes an array of `int` as a parameter and returns the mode of the array as an `int`. The method assumes that the values in the array are sorted in increasing order. When looking for the *mode*, the method **SHOULD** take benefit of the fact that the numbers are ordered and same numbers are stored in consecutive locations. A method that returns the correct *mode*, but does not utilize this fact will be awarded a maximum grade of 6 / 10. Assume that the method is `public` and `static`.

3. (10 points) Consider the following program:

```
1  public class StackTrace {
2     public static char m1(String s, int i) {
3        int x;
4        int y;
5        int z;
6        char c;
7
8        x = i + 4;
9        if (m2(x)) {
10          y = x * 2;
11          // Point #1
12          z = m3(y);
13          // Point #2
14          c = s.charAt(z);
15       } else {
16          c = s.charAt(i);
17       }
18       return c;
19    }
20
21    public static boolean m2(int x) {
22       int a;
23       int b;
24
25       a = 3;
26       b = 25;
27       // Point #3
28       return ((a * x) > b);
29    }
30
31    public static int m3(int x) {
32       int a;
33       int b;
34
35       a = 2;
36       b = 6;
37       // Point #4
38       return x / a + b;
39    }
40
41    public static void main(String[] args) {
42       int index;
43       char character;
44       String text;
45
46       text = "This question involves the run-time stack";
47       index = 5;
48       character = m1(text, index);
49       // Point #5
50       System.out.println(character);
51    }
52 }
```

Trace the execution of the above program, and write the contents of the run-time stack every time the

execution encounters one of the points listed in the above program in comments. That is, if a code fragment contains a comment like this one:

```
// Point #0
```

then whenever program execution reaches this comment, you should write the contents of the run-time stack at that point. Use the line numbers to the left of each statement as return addresses.

You **MUST** write the contents of the run-time stack in the order in which the points in comments are reached by the program's execution, not in the order in which they are listed in the above program source code. Therefore, if execution reaches point #4 before point #2, the contents of the stack at point #4 **MUST** be written before the contents of the stack at point #2.

4. (10 Points) Write a RECURSIVE method called `gcd()` that computes the greatest common divisor (GCD) of two integers. The method accepts as input two `int` parameters `a` and `b`, and returns an `int` that represents the GCD of the two numbers. The greatest common divisor of two numbers *a* and *b* is defined as the largest number that divides both *a* and *b*.

Here is the method declaration:

```
public int gcd (int a, int b)
{
      // fill in this method
}
```

Examples:

```
gcd(4, 4) = 4;
gcd(6, 9) = 3;
gcd(0, 5) = 5;
gcd(5, 0) = 5;
gcd(13, 4) = 1;
```

You can use the following mathematical properties of GCD:

- The GCD of two integers *a* and *b* is equal to the GCD of *b* and the remainder of the division of *a* by *b* (if *a* is the bigger of the two integers). For example, the GCD of 12 and 8 is equal to the GCD of 8 and 4, where 4 is the remainder when you divide 12 by 8.
- The GCD of any integer *x* and 0 is *x*, because 0 is divisible by anything.

You can also assume that both the parameters `a` and `b` are always going to be non-negative.

Section 3: Programming (10 + 20 + 10 points; 40 points total)

This section is divided into **three programming problems**. Unlike section 2, these three problems make up one complete program and you must program it in its entirety. Make sure to follow the instructions carefully. You may use the Java library reference appended to the questionnaire for help.

The complete program reads and displays information about students registered in COMP-202. This information includes assignment grades (including bonus question grades), midterm grades, and final exam grades for a number of COMP-202 students. This program consists of 3 classes:

- A `COMP202Student` class; objects which belong to this class represent COMP-202 students and their respective grades.
- A `StudentLoader` class, which defines a method called `load()` that loads (reads) the information stored in a file about COMP-202 students so this information can be manipulated by the program.
- A `GradeLister` class, which defines a `main()` method that utilizes both the `COMP202Student` and `StudentLoader` classes to read grade information for all the students from a given file, compute the final grade for each student, and display the information on the screen in a nicely formatted way.

1. Complete the definition of the `COMP202Student` class by writing a key computational method (10 points)

Part of the `COMP202Student` class is already written. The class contains attributes and methods as defined below:

```
public class COMP202Student {
   private String firstName;
   private String lastName;
   private long id;

   private int[] assignments;     // assignment grades (each out of 100)
   private int[] bonuses;         // bonus question grades (each out of 1)
   private int midtermExam;       // midterm grade (out of 100)
   private int finalExam;         // final exam grade (out of 100)

   private double finalGrade;     // final course grade (out of 100)

   public COMP202Student(String lastName, String firstName,
                         long id, int[] assignments, int[] bonuses,
                         int midtermExam, int finalExam){
      // The body of this constructor is already written
   }

   public int compareTo(COMP202Student otherStudent){
      // The body of this method is already written
   }

   private void computeFinalGrade(){
      //*** YOU MUST WRITE THIS METHOD ***
   }
```

```
   // The following methods are defined to access private attributes
   public long getID(){
      return this.id;
   }

   public String getLastName(){
      return this.lastName;
   }

   public String getFirstName(){
      return this.firstName;
   }

   public double getFinalGrade(){
      return this.finalGrade;
   }
}
```

You MUST write ONLY the complete definition of the `computeFinalGrade()` method in your exam booklet. Assume that the bodies of the constructor, the `compareTo()` method and four other accessor methods have already been written.

The constructor initializes all the attributes of the new `COMP202Sudent` object. After initializing all the attributes except `finalGrade` from the parameters, it invokes the helper method `computeFinalGrade()` to compute and set the value of the `finalGrade` attribute.

The `compareTo()` method compares the `COMP202Student` object it is called on to another `COMP202Student` object `otherStudent` passed as parameter, based on their full names. This method returns a negative number, `0`, or a positive number if the full name of this `COMP202Student` object occurs before, is equal, or occurs after the full name of the `otherStudent` object, respectively. In other words, to obtain the result of the comparison, the `compareTo()` method compares the last names of the `COMP202Student` objects, and if they are equal, it then compares their first names.

The `computeFinalGrade()` method computes the final grade of a COMP-202 student and assigns this value to the instance variable named `finalGrade`. The final grade of a COMP-202 student is computed according to the following formula:

$$F = 0.5 * X + 0.3 * A + 0.2 * M$$

In the above formula, F represents the final grade, X represents the final exam grade, A represents the assignment average, and M represents the midterm grade. The values of all these variables are within the range 0-100, inclusively.

Remember that the lowest assignment grade is not counted in the average. Thus, if there are 10 assignments, only the 9 best assignment grades are included in the assignment average.

The bonus points affect the final grade in the following manner:

- 0-2 bonus points: No change

- 3-4 bonus point: Final grade increases by 1
- 5 bonus points: Final grade increases by 2
- 6 or more bonus points: Final grade increases by 3

Your `computeFinalGrade()` method **MAY** assume that variables `assignments`, `bonuses`, `midtermExam`, and `finalExam` have already been initialized when the `computeFinalGrade()` method is called on a `COMP202Student` object. It **MAY** also assume that there are exactly as many bonus questions as there are assignments. However, your `computeFinalGrade()` method **MUST NOT** assume that there are only 6 assignment and bonus question grades; in other words, it **MUST** work for any number of assignments and bonus questions.

2. Define the `StudentLoader` class (20 points)

You must write the complete definition of the `StudentLoader` class in your exam booklet.

The class contains **only one `static` method named `load()`**, which opens the file whose path is given by a `String` parameter `filename`, and returns an `ArrayList` of `COMP202Student` objects created with the information read from this file.

The text file containing information about COMP-202 students is already created. Inside the file, the information associated with each student is listed on a different line. Within a line, the information is listed in the following order:

- The student's last name
- The student's first name
- The student's ID number
- Pairs of numbers representing the student's assignment and bonus question grade; there are 6 such pairs
- The midterm exam grade
- The final exam grade

In a single line, these values are separated by one or more white space characters (space or tab). Note that there might be zero or more white space characters on a line before the student's first name or after his / her final exam grade.

For example, if student Archibald Haddock has student ID number 123456789, received grades of 100 on all of his assignments, 1 on all his bonus questions, 95 on his midterm and 98 on his final, his entry in the file will look like this:

```
Haddock Archibald 123456789 100 1 100 1 100 1 100 1 100 1 100 1 95 98
```

Notes:

- The `COMP202Student` objects within the `ArrayList` **MUST** be sorted in alphabetical order of their full names when the `ArrayList` is returned. You can compare two `COMP202Student` objects using the `compareTo()` method defined in `COMP202Student` class.
- When opening the file, a `FileNotFoundException` could occur. Your `load()` method **MUST**

propagate this exception object to the method which called `load()`.

3. The `GradeLister` class and the `main` method (10 points)

You MUST write the complete definition of the `GradeLister` class in your exam booklet.

The `GradeLister` class contains only the `main()` method. The header of the of the `main()` method is as usual. This `main()` method accepts exactly one command-line parameter, and this parameter represents the path of a file that contains student information in the format described in part 2. Your program **MUST** load the information stored in this file (by calling the `load()` method described in part 2), and display the following information about all the students, one student per line. Each line should show the following information for a student, separated by a tab character, in the order this information is listed:

- The student's last name
- The student's first name
- The student's ID number
- The student's final grade

To retrieve these attributes from a `COMP202Student` object, you may use the methods already defined in the `COMP202Student` class.

Notes:

- If no command-line parameters are passed to the `main()` method, or more than one command line parameter is passed to the `main()` method, the latter should display an error message and terminate.
- If a `FileNotFoundException` occurs when the `main()` method calls the `load()` method, the `main()` method **MUST** catch the exception object, display an error message, and terminate.

Appendix A: Java Standard Library Reference

Package: java.lang

class String

- int compareTo(String s)
- boolean equals(String s)
- boolean equalsIgnoreCase(String s)
- char charAt(int index)
- int length()
- String substring(int begin, int end)
- String toUpperCase()
- String trim()

class Integer

- static int parseInt(String s)

class Double

- static double parseDouble(String s)

class Character

- static boolean isDigit(char ch)
- static boolean isLetter(char ch)
- static boolean isUpperCase(char ch)
- static boolean isWhitespace(char ch)

Package: java.util

class Scanner

- `Scanner(File source) throws FileNotFoundException`
- `Scanner(InputStream source)`
- `Scanner(String source)`
- `boolean hasNext()`
- `String next()`
- `double nextDouble()`
- `float nextFloat()`
- `int nextInt()`
- `long nextLong()`
- `void close()`

class ArrayList

- `boolean add(Object element)`
- `void add(int index, Object element)`
- `void clear()`
- `Object get(int index)`
- `boolean isEmpty()`
- `int indexOf(Object elem)`
- `Object remove(int index)`
- `int size()`

Package: java.io

class File

- `File(String pathname)`
  (Constructor) Creates a new `File` instance by from the file/pathname given as a `String`.

class PrintStream

- `PrintStream(File file) throws FileNotFoundException`
- `PrintStream(OutputStream out)`
- `PrintStream(String fileName) throws FileNotFoundException`
- `void print(Object obj)`
- `void println(Object x)`
- `void close()`